UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Secuenciación de líneas de ensamblaje mixtas con estaciones abiertas mediante un algoritmo de enumeración

Gerrit Faerber, Anna Maria Coves Moreno

IOC-DT-P-2004-10
Juny 2004

Institut d'Organització i Control
de Sistemes Industrials

Instituto de Organización y control de Sistemas Industriales (IOC)

Programa Doctorado: Automatización Avanzada y Robótica
Optimización Combinatoria

Gerrit Färber
Dra. Anna Maria Coves Moreno

Secuenciación en líneas de ensamblaje mixtas con estaciones abiertas mediante un algoritmo de enumeración

RESUMEN

El objetivo de este trabajo es el diseño y la aplicación de un algoritmo de enumeración para hallar la longitud óptima de cada estación, junto con la secuencia asociada de modo que se minimice tanto el tiempo ocioso (Idle-time) de los operadores en una línea de ensamblaje mixta. La línea de ensamblaje tiene una longitud y un número definido de estaciones. El Software diseñado se ha implementado en C++ y utiliza una búsqueda a lo ancho. Se presenta además un método rápido para conseguir todas las secuencias posibles, utilizando un sistema numérico con la base igual al número de piezas para producir. El algoritmo de enumeración es muy útil siempre y cuando la dimensión del problema sea modesta.

ABSTRACT

The objective of this work is the design and also the application of an enumeration algorithm for a mixed assembly line in order to find the optimal length of each station, as well as the associated sequence so that the Idle-time of the operators is minimal. The assembly line consists of a total length and a defined number of stations. The designed Software is implemented in C++ and utilises a "Breath-First Search" (BFS). Furthermore a fast method is presented for the determination of all possible sequences, utilizing a numerical system with the base equal to the number of pieces to produce. The enumeration algorithm is very useful as long as the dimension of the problem is modest.

Palabras clave: Mixed model assembly line, Algoritmo de enumeración

INDICE

1. INTRODUCCIÓN

En los últimos años aumenta cada vez más la necesidad de producir diferentes tipos de productos en una misma línea de producción. Este tipo de producción se llama producción mixta. Existe la necesidad de líneas mixtas cuando la demanda de cada uno de los productos es pequeña, la línea en este caso debe ser construida de tal manera que sea más flexible, y permita la producción de los diferentes productos.

Los productos pueden variar mucho entre ellos o solamente variar de tal manera que necesitan distintos módulos, por ejemplo, en un armario dos puertas grandes en vez de tres pequeñas.
Este tipo de producción se encuentra cada vez más en la industria, por la necesidad de ofrecer una variación de productos más grande al cliente. Un ejemplo de este tipo de producción puede verse en la industria de automóviles. Aunque la fabricación es en serie, las empresas ofrecen al cliente que elija entre una lista de componentes opcionales, como tipo de motor, color, etc. El coche se fabrica según el deseo del cliente. La línea tiene que permitir una producción con muchos modelos. Además de ofrecer un mejor servicio al cliente, el stock de productos acabados disminuye notablemente y con ello los gastos de almacén.

La desventaja de este modelo mixto es la mayor complejidad en la organización y la infraestructura necesaria para soportar la producción. Los operadores deben tener más experiencia y ser más flexibles, por otra parte, el trabajo resulta más interesante y más atractivo.
En las referencias Sarker and Pan, 1997, y Sarker and Pan, 2001, los autores definen una línea de producción mixta con tres estaciones y determinan las fórmulas para poder calcular la línea mediante un modelo de programación entera mixta. En la referencia Kim and Jeong, 2000, los autores describen un algoritmo de "Branch & Bound" para obtener la optimización de "Unfinished work".
El trabajo que se presentará en el presente documento utiliza una variación del algoritmo descrito en Kim and Jeong, 2000, para el problema planteado en Sarker and Pan, 1997, y Sarker and Pan, 2001.

2. DESCRIPCIÓN DE LA LÍNEA

En una línea de ensamblaje se puede fabricar varios modelos de uno o más productos. Cada uno de estos modelos necesita un tiempo de operación distinto en cada estación. Estos tiempos están definidos en una matriz de tiempo de ensamblaje. La siguiente imagen (imagen-1) muestra una cinta con cuatro modelos diferentes de armarios.

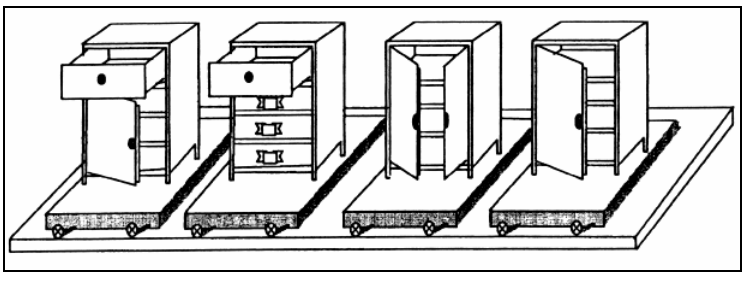


Imagen-1: Producción mixta: Línea de ensamblaje de modelos mixtos
con 4 distintos armarios. Extraído de Sarker and Pan, 2001.

El ensamblaje de un producto está asignado a distintas estaciones, en las que suponiendo que trabaja un solo operador. Existen dos tipos de estaciones, la estación cerrada y la estación abierta, la diferencia es que en la estación abierta, el operador cuando termina su trabajo con una cierta pieza puede pasar la frontera de la estación que tiene asignada e ir donde hay otro operador. En estaciones cerradas, el operador no puede pasar dichas fronteras.

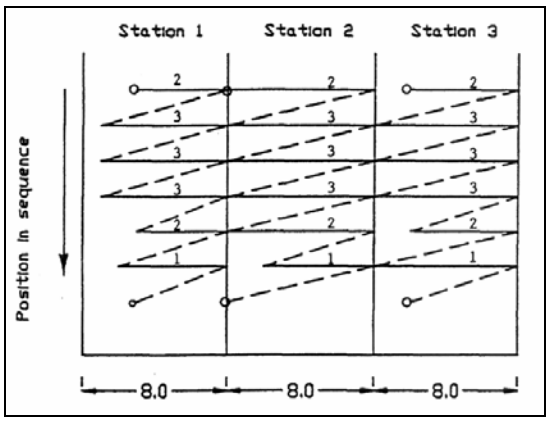


Imagen-2a: Estaciónes cerradas

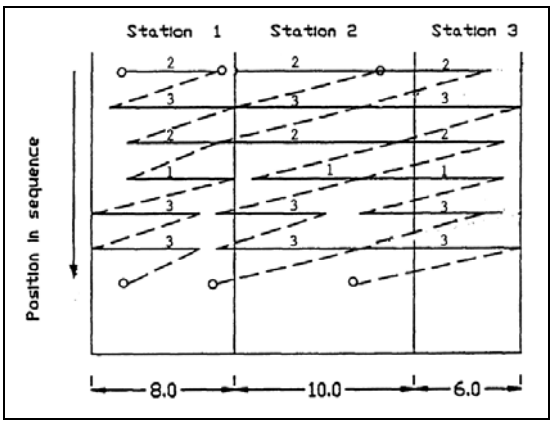


Imagen-2b: Estaciónes abiertas a la izquierda

Imagen-2: Ejemplo de los movimientos de los operadores en una línea con estaciones cerradas (2a) y estaciones abiertas a la izquierda (2b). La primera estación siempre es una estación cerrada. Extraído de Sarker and Pan, 1997.

En el programa que se describe en el presente documento se implementa el sistema de estaciones abiertas a la izquierda. Esto significa que el operador puede empezar a trabajar en la estación anterior, pero tiene que terminar su trabajo cuando llega a la frontera derecha. En caso de no terminar el trabajo asignado a la estación un operador auxiliar acude a finalizar el trabajo durante el tiempo necesario auxiliar (Utility-time). El operador de la siguiente estación no puede empezar a trabajar con esta pieza hasta que el trabajo asignado a la estación anterior haya finalizado. La primera estación es una estación cerrada, por lo tanto, en el cálculo hay diferentes fórmulas, que dependen de si es la primera estación o no.

La imagen-2b nos muestra el resultado de una línea con estaciones abiertas a la izquierda. La línea se forma con tres estaciones con las longitudes 8, 10 y 6 respectivamente. Las líneas contínuas muestran el desplazamiento de las piezas en la línea de producción. Cuando estas líneas se interrumpen, las piezas siguen desplazándose en la línea de producción pero ningún operador se dedica a trabajar con ellas. Las líneas discontínuas muestran el recorrido del operador desde el momento en que termina con una pieza y vuelve donde tiene que empezar con la siguiente pieza. En la imagen–2b se encuentra la secuenciación de piezas 2-3-2-1-3-3. En esta imagen, observando la quinta pieza que es de modelo 3, vemos que el operador de la estación 1 termina con la pieza antes de llegar al final de su estación y que el operador de la estación 2 traspasa la frontera izquierda y empieza con la pieza en la estación 1. Observando todas las piezas podemos ver que el operador de la segunda estación invade la primera estación en tres ocasiones, para la quinta pieza ya comentada y para la tercera y la sexta. También el operador de la estación 3 invade la estación 2 en este caso para todas las piezas.

3. ASIGNACIÓN DE VARIABLES

Para poder calcular la longitud asignada de las estaciones y la secuencia óptima se define las siguientes variables:

Índices;

i Estación, $i \in \{1,2,...,N\}$

j Pieza, $j \in \{1,2,...,K\}$

m Modelo $m \in \{1,2,...,M\}$

Parámetros de entrada:

N Número de estaciones

K Número total de piezas para producir

M Número de modelos en Minimal-Part-Set

L_{max} Longitud máxima de la línea

$t_{m,i}$ Tiempo de ensamblaje del modelo *m* en la estación *i*

V_C Velocidad de la línea

V_0 Velocidad de los operadores ($V_C >> V_0$)

$Z_{i,j}$ Starting-Point (Punto de inicio) de un operador en la estación *i* para la pieza *j*

Z_i^0 Punto de referencia de la estación *i* coincide con el inicio de la estación

Parámetros de salida:

$d_{i,j}$ Upstream-Distance distancia recorrida aguas arriba de un operador de la estación *i* para iniciar el trabajo con la la pieza *j*

$I_{i,j}$ Idle-Time (Tiempo ocioso) de un operador en la estación *i* para la pieza *j*

L_i Longitud de la estación *i*

$u_{i,j}$ Utility-Time (Tiempo auxiliar) de un operador auxiliar en la estación *i* para la pieza *j*

$X_{m,j}$ 1 si la pieza *j* es del modelo *m*, 0 si no lo es

λ Launch-Time, Intervalo entre dos piezas en la línea (constante)

4. DEFINICIONES Y FORMULAS

Las siguientes fórmulas están aplicadas en el cálculo de la línea con estaciones abiertas a la izquierda. No obstante, la primera de las estaciones se comporta como una estación cerrada tal y como ya hemos comentado anteriormente. El primer operador no puede empezar antes de su punto de referencia (Z_1^0).

Por lo tanto, el que Starting-Point, Idle-Time y Upstream-Distance tengan dos formulas, dependerá de si es la primera estación o no[1].

Los datos de partida son la longitud total de la línea, la matriz de tiempo de ensamblaje y también el número de piezas de cada modelo que deben ser producidas, por ejemplo: de seis piezas que salen de la producción, uno tiene que ser del modelo 1, dos del modelo 2 y tres del modelo 3. Esta definición se encuentra en el vector "Minimal-Part-Set", que para este ejemplo será (1, 2, 3).

Las expresiones que se describen a continuación han sido extraídas de Sarker and Pan, 1997 y Sarker and Pan, 2001.

4.1 Starting-Point: (Punto de inicio)

El lugar donde un operador empieza a trabajar con una cierta pieza es el *Starting-Point*. En una estación abierta a la izquierda el Starting-Point puede ser distinto a la frontera izquierda, donde el operador anterior tiene que terminar su trabajo.

Primera estación:

$$\boxed{Z_{1,j} \geq Z_1^0}$$

con $j \in \{1,2,...,K\}$

Otras estaciones:

$$\boxed{Z_{i,j+1} = Z_{i,j} + v_c \cdot \sum_{m=1}^{M} (X_{m,j} \cdot t_{m,i}) - v_c \cdot u_{i,j} - d_{i,j}}$$

con $i \in \{2,3,...,N\}$ y $j \in \{1,2,...,K-1\}$

Condición:

$$\boxed{Z_{i+1,j} \geq Z_{i,j} + v_c \cdot \sum_{m=1}^{M} (X_{m,j} \cdot t_{m,i})}$$

con $i \in \{1,2,...,N-1\}$ y $j \in \{1,2,...,K\}$

4.2 Utility-Time: (Tiempo auxiliar)

En el caso de que un operador no termine el trabajo con una cierta pieza y llegue a la frontera derecha, tiene que terminar y volver a trabajar con la pieza siguiente. En este caso un operador auxiliar tiene que venir y terminar con esta pieza. El tiempo que necesita el operador auxiliar, es el *Utility-Time*.

Todas estaciones:

$$\boxed{u_{i,j} = MAX\left\{\sum_{m=1}^{M} (X_{m,j} \cdot t_{m,i}) - \frac{(L_1 + L_2 + \cdots + L_i) - Z_{i,j} + Z_1^0}{v_c}, 0\right\}}$$

con $i \in \{1,2,...,N\}$ y $j \in \{1,2,...,K\}$

4.3 Idle-Time: (Tiempo ocioso)

Si al contrario, un operador termina su trabajo con una cierta pieza y no puede empezar con la siguiente, porque el último operador no ha terminado con ésta, el primer operador tendrá que esperar. Este tiempo se llama *Idle-Time*.

Primera estación:

$$\boxed{I_{1,j+1} = MAX\left\{0, \lambda - \left[\frac{Z_{1,j} - Z_1^0}{v_c} + \sum_{m=1}^{M} (X_{m,j} \cdot t_{m,i}) - u_{1,j}\right]\right\}}$$

con $j \in \{1,2,...,K-1\}$

Otras estaciones:

$$\boxed{I_{i,j+1} = MAX\left\{0, \lambda - \frac{d_{i,j}}{v_c}\right\}}$$

con $i \in \{2,3,...,N\}$ y $j \in \{1,2,...,K-1\}$

[1]Estos términos se definen en las expresiones correspondientes.

4.4 Upstream-Distance: (Distancia aguas arriba)

El operador se desplaza acompañando la línea mietras trabaja, a la misma velocidad que también se desplaza la línea. Cuando termina su trabajo con la pieza, el operador vuelve hasta el punto donde tiene que empezar con la siguiente pieza. La distancia que entonces recorre es la *Upstream-Distance*.

Primera estación:

$$d_{1,j} = MIN\left\{ Z_{1,j} - Z_1^0 + v_c \cdot \sum_{m=1}^{M} (X_{m,j} \cdot t_{m,i}) - v_c \cdot u_{1,j}, \lambda \cdot v_c \right\}$$

con $i \in \{1,2,...,N\}$

Otras estaciones:

$$d_{i,j} = MIN\left\{ Z_{i,j} + v_c \cdot \sum_{m=1}^{M} (X_{m,j} \cdot t_{m,i}) - v_c \cdot u_{i,j} - Z_{i,j+1}, \lambda \cdot v_c \right\}$$

con $i \in \{2,3,...,N\}$ y $j \in \{1,2,...,K-1\}$

5. ALGORITMO DE ENUMERACIÓN

El programa implementado tiene como objetivo minimizar el Utility-Time y el Idle-Time, para encontrar la longitud óptima de cada estación y la mejor secuencia. La función objetivo a minimizar en el presente trabajo es una combinación lineal de los dos tiempos usando un peso relacionado el coste de cada estación. El procedimiento que se ha utilizado y que se describe en el presente trabajo es:

1. Se leen los datos de entrada de un archivo.
2. Se calcula un coste inicial.
3. Se genera el árbol de la secuencia con un mínimo de ramas.
4. Se diseña un algoritmo de enumeración con un árbol de tantos niveles como estaciones, aplicando una búsqueda a lo ancho.

5.1 Tabla de base de datos

La tabla (Tabla-3) contiene los datos base para el algoritmo de enumeración. El número de estaciones es tres y el número de modelos en la línea mixta es tres, según el "Minimal-Part-Set" hay seis piezas para producir (1+2+3 = 6).

Input parameters	Notation	Value	Unit
Number of stations	N	3	-
Number of models	M	3	-
Minimal part set	D_m	(1, 2, 3)	Items
Station 1	t_{m1}	(6, 5, 7)	Minutes
Station 2	t_{m2}	(6, 10, 8)	Minutes
Station 3	t_{m3}	(8, 6, 9)	Minutes
Unit idle-time cost	C_i^I	(0.2, 0,2, 0,2)	Dollar/min
Unit utility-time cost	C_i^U	(0.5, 0,5, 0,5)	Dollar/min
Line length	L_{max}	24	Feet
Conveyor speed	V_C	1	Feet/min
Operator speed	V_O	∞	Feet/min

Tabla-3: Tabla de base de datos para el cálculo de la línea mixta. Extraído de Sarker and Pan, 1997.

5.2 Cálculo del coste inicial

La secuencia inicial para calcular el coste inicial depende del "Minimal-Part-Set". Es decir, primero se obtiene solamente modelos del tipo uno, tantos como están definidos en el "Minimal-Part-Set". Después se obtiene solamente piezas del tipo dos etc. Para determinar las longitudes iniciales de cada estación, se divide la longitud máxima por el número de las estaciones, es decir inicialmente se consideran todas las estaciones de igual longitud.

El coste inicial se determina con los valores del Idle-time y Utility-time de la configuración inicial y los correspondientes pesos. Este coste es el coste superior y se utilizará para destacar soluciones que son peores en árbol de enumeración. En vez de dividir la longitud máxima de la línea entre el número de estaciones se puede utilizar una heurística para encontrar una solución inicial.

El trabajo de Kim and Jeong, 2000, optimiza la línea de producción mediante un algoritmo de "Branch & Bound". En la línea de producción se utiliza el Setup-time que es el tiempo necesario para cambiar de un modelo a otro. Ese trabajo utiliza una heurística para encontrar la secuencia con los valores mínimos del Setup-time.

5.3 Árbol de Secuencia

El hecho de que los tiempos no varían mucho entre si hace que al ir secuenciando las piezas, según las ramas del árbol, no puedan descartarse secuencias comparadas con el coste inicial hasta prácticamente al final del árbol. Por lo tanto utilizando un árbol con un algoritmo de enumeración calcularía casi todos los caminos hasta el final, el algoritmo de enumeración está aplicado solamente para las longitudes de las estaciones. Se calculan todas las secuencias.

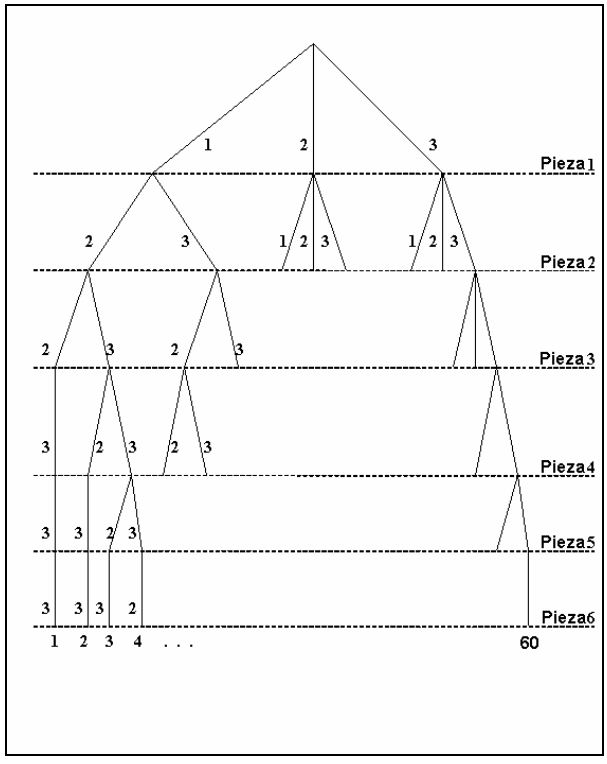


Imagen-4: Árbol de la secuencia, minimizado a 126 cálculos.

El árbol contendrá tantos niveles como piezas a fabricar, en nuestro caso 6. La determinación de todas las secuencias posibles se realiza mediante un sistema numérico de base seis. En este sistema el número mínimo es 000000. Incrementando este número hasta el número 543210, todas las posibilidades son calculadas. Se tienen que cambiar las cifras: el 0 por "1", el 1 y 2 por "2" y el 3, 4 y 5 por "3". Después es necesario eliminar las soluciones dobles que resultan en el cambio de las cifras. El número máximo de secuencias se puede comprobar con la siguiente ecuación, extraído de Korkmazel and Meral, 2001:

$$\frac{K!}{k_1!\cdot k_2!\cdot\ldots\cdot k_M!}$$

En total quedan 6!/(1!·2!·3!) = 60 secuencias posibles para calcular (imagen-4).

5.4 Construcción del árbol inicial para las longitudes de las estaciones.

El árbol inicial contiene la siguiente información en cada vértice:

> Número del vértice, Nivel, Número de nodo, Antecesor, Utility-Time, Idle-Time, Coste, Longitud de la estación, Longitud total, Número de descendientes, Indicador (si tiene sentido continuar con un cierto descendiente).

La Tabla–5 muestra los datos guardados para cada vértice que aparece en la imagen-6 del árbol del algoritmo de enumeración. Los "Números de vértices" son los números continuos de los vértices que pertenecen al mismo antecesor. El "Nivel" determina la estación, se ha introducido el nivel 0 como un nivel virtual que incluye los 22 primeros vertices. Los "Numeros de nodo" son números contínuos de los vértices hasta el último vértice en el último nivel. El "Antecesor" es el "Numero de nodo" del antecesor.

Structure Node									
Numero del vertice	1								
Nivel	1								
Numero de nodo	1								
Antecesor	0								
Utility-Time	99999								
Idle-Time	99999								
Coste	99999								
Longitud de estación	1								
Longitud total	1								
Numero de descendiente	23	24	25	26	27	28	28	.....	44
Indicador para descendiente	0	0	0	0	1	1	1	.....	0

Tabla-5: Definición de un nodo del algoritmo de enumeración.

Con los valores de "Utility-Time" y "Idle-Time" se calcula el "Coste", el coste de estos vértices. Al principio el coste tiene un valor inicial de 99999. La "Longitud de estación" es la longitud de la estación de dicho vértice y la "Longitud total" es la longitud que se calcula teniendo en cuenta los antecesores hasta nivel-0. En nuestro ejemplo este número debe ser 24 para cada vértice en el nivel más bajo.

Los “Números de descendientes” y los “Indicadores para descendientes” forman una matriz que contiene los “Números de vértice” de todos los descendientes de este vértice. Su correspondiente indicador 0 o 1 depende de si tiene sentido bajar al vertice del descendiente. Al principio se define un árbol inicial que tiene en cada nivel tantas ramas como longitudes con valor entero haya.

5.5 Reducción del árbol para encontrar soluciones factibles

El árbol, explicado en 5.4, todavía tiene caminos no factibles. Por ejemplo las longitudes 10+10+10 no son factibles si la longitud total es 24. Eliminando todas las soluciones no factibles se queda un árbol más pequeño con soluciones que al principio son factibles. El árbol inicial tiene 10648 (“Numero de vertices” elevado por estaciones = 22^3) soluciones por calcular. Con la reducción quedan solamente 253 soluciones factibles (ver imagen-6). Sin embargo, el algoritmo de enumeración al principio tiene más de 253 soluciones por calcular, porque en cada nivel tiene que hacer varios cálculos. La imagen-6 muestra el árbol resultante con los caminos al principio factibles.

5.6 Exploración en el Algoritmo de Enumeración

La exploración del árbol, explicado en 5.5, empieza con el cálculo del primer nivel. Se calculan solamente los tiempos de Idle-time y Utility-time que resultan para la secuencia elegida hasta la estación uno. Con estos tiempos y los correspondientes pesos de la estación uno se calculan los costes de los nodos en el primer nivel. En el caso de que el coste calculado de un nodo no sea mejor que el coste inicial, pondremos a cero los indicadores de sus descendientes. En ese caso sus descendientes ya no estarán calculados. Si el coste es mejor que el coste inicial no podemos cambiar el coste inicial porque el cálculo del coste de este nodo tiene en cuenta solamente las estaciones hasta este nivel.

Al bajar de nivel, el indicador del antecesor se comprueba antes de hacer el cálculo de este nodo, así que se calculará un nodo solamente si su indicador está en cero. El cálculo se hace a lo ancho, lo que significa que en cada nivel se calculan primero todas las soluciones factibles antes de bajar de nivel.

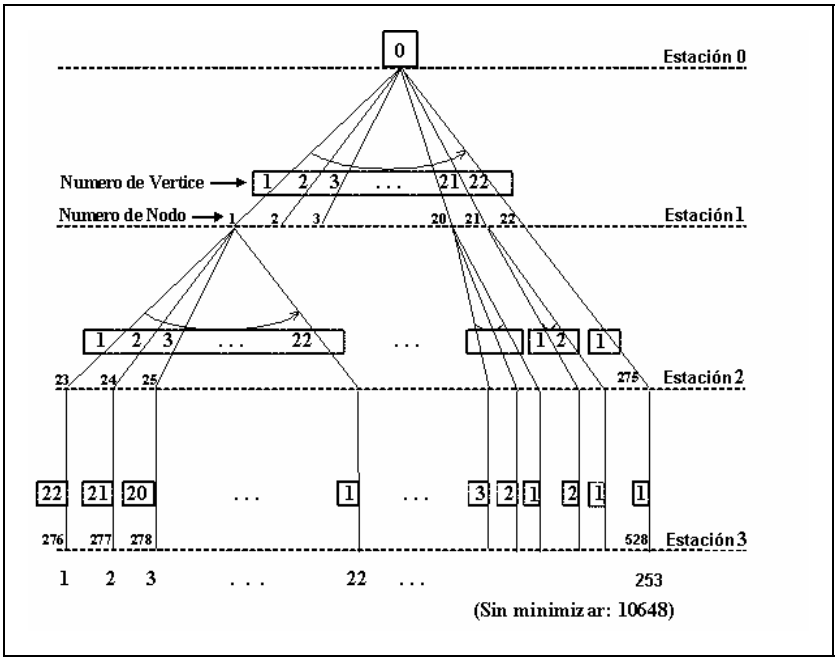


Imagen-6: Árbol reducido del algoritmo de enumeración.

6. SOFTWARE

El programa implementado y que recoge el procedimiento descrito en los anteriores apartados se ha denominado “MixMod.exe”. Los datos iniciales que figuran en la Tabla-3 pueden ser modificados deben ser archivados en la misma carpeta que el programa ejecutable.

Se accede a los datos de un archivo mediante el nombre “Input.txt” (ver imagen-7) que tiene que estar en la misma carpeta como el programa ejecutable. Las entradas del programa son las mismas que se puede encontrar en la Tabla-3.

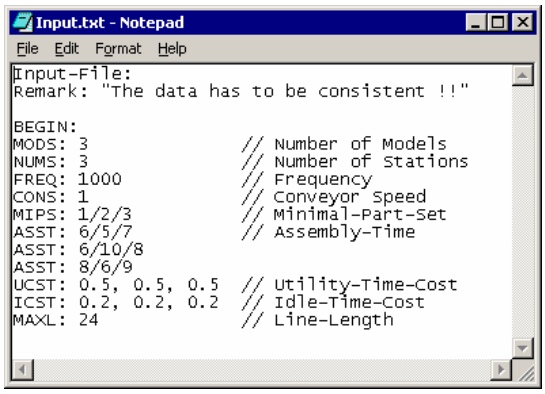


Imagen-7: Formato de datos de entrada.

El programa empieza con la ventana “MixMod” (ver imagen-8). Se puede variar el Launch-Time y también el archivo que contiene los demás parámetros. La ventana tiene dos barras de progreso para dar una idea cuanto tiempo va a durar el cálculo.

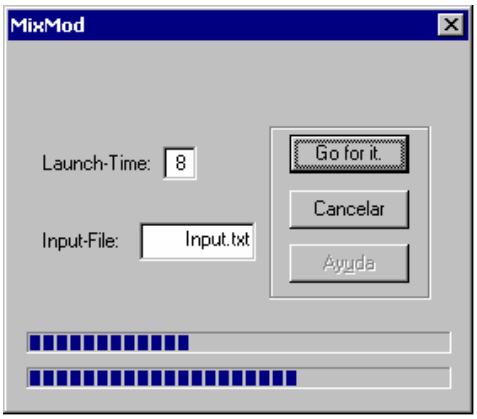


Imagen-8: Ventana principal del Programa “MixMod”.

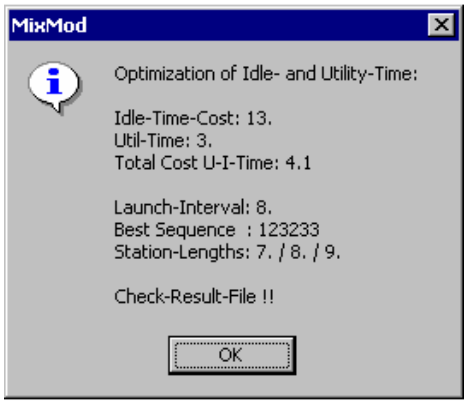
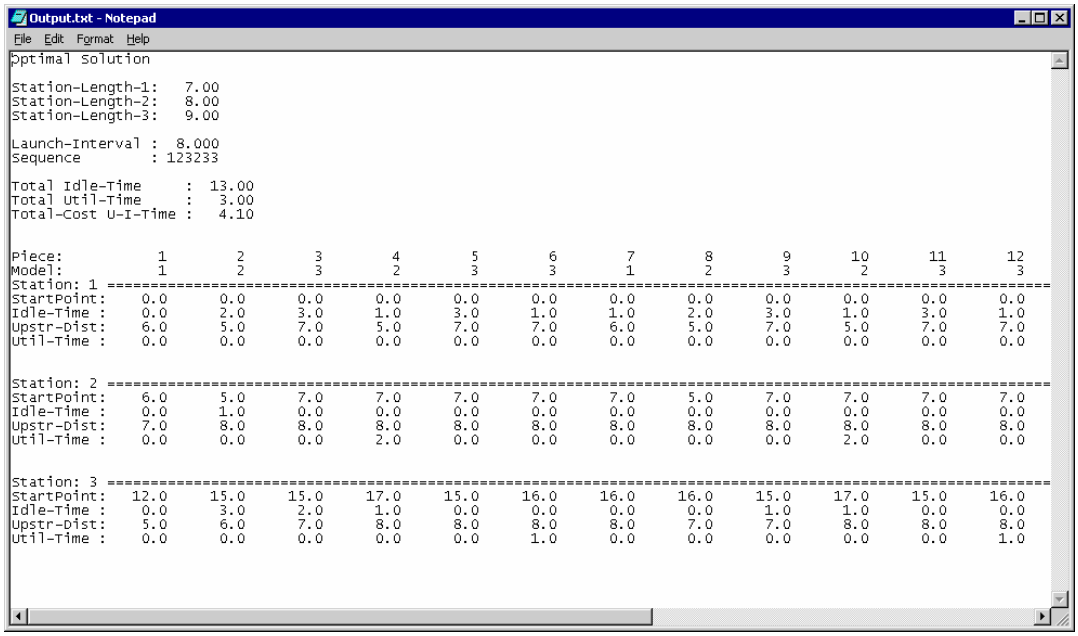


Imagen-9: Ventana del resumen de la Optimización.

Al final cuando el programa termina muestra un resumen de los resultados (imagen-9). Primero muestra que se ha optimizado el Idle- y el Utility-time. Luego muestra los valores de Idle-Time, Utility-Time y el coste total de la solución, la secuencia encontrada y las longitudes optimas de las estaciones.

Los datos detallados están grabados en el archivo “output.txt” (imagen-10). Se muestra una tabla con datos de “Starting-Point”, “Idle-Time”, “Utility-Time” and “Upstream-Distance”de las estaciones para cada pieza. Con esta tabla se puede planificar las longitudes de la línea.
La imagen-10 muestra la solución óptima de la línea según los valores del archivo “Input.txt” mostrado en la imagen-7. La secuenciación óptima es 1-2-3-2-3-3 y las longitudes de las estaciones son 7, 8 y 9. El Idle-Time resultante es 13.


Output.txt - Notepad

File Edit Format Help

```
Optimal Solution

Station-Length-1:   7.00
Station-Length-2:   8.00
Station-Length-3:   9.00

Launch-Interval :  8.000
Sequence        : 123233

Total Idle-Time     :  13.00
Total Util-Time     :   3.00
Total-Cost U-I-Time :   4.10


Piece:           1       2       3       4       5       6       7       8       9      10      11      12
Model:           1       2       3       2       3       3       1       2       3       2       3       3
Station: 1 ==========================================================================================================
StartPoint:    0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
Idle-Time :    0.0     2.0     3.0     1.0     3.0     1.0     1.0     2.0     3.0     1.0     3.0     1.0
Upstr-Dist:    6.0     5.0     7.0     5.0     7.0     7.0     6.0     5.0     7.0     5.0     7.0     7.0
Util-Time :    0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0


Station: 2 ==========================================================================================================
StartPoint:    6.0     5.0     7.0     7.0     7.0     7.0     7.0     5.0     7.0     7.0     7.0     7.0
Idle-Time :    0.0     1.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0
Upstr-Dist:    7.0     8.0     8.0     8.0     8.0     8.0     8.0     8.0     8.0     8.0     8.0     8.0
Util-Time :    0.0     0.0     0.0     2.0     0.0     0.0     0.0     0.0     0.0     2.0     0.0     0.0


Station: 3 ==========================================================================================================
StartPoint:   12.0    15.0    15.0    17.0    15.0    16.0    16.0    16.0    15.0    17.0    15.0    16.0
Idle-Time :    0.0     3.0     2.0     1.0     0.0     0.0     0.0     0.0     1.0     1.0     0.0     0.0
Upstr-Dist:    5.0     6.0     7.0     8.0     8.0     8.0     8.0     7.0     7.0     8.0     8.0     8.0
Util-Time :    0.0     0.0     0.0     0.0     0.0     1.0     0.0     0.0     0.0     0.0     0.0     1.0
```

Imagen-10: Archivo “Output.txt” con los datos detallados.

Por ejemplo, en imagen-10 el operador de la primera estación, cuando termina con la cuarta pieza (modelo 2), tiene que correr 5 unidades hacia arriba y esperar 3 minutos hasta que llegue la pieza 5 (modelo 3) por que el tiempo de ensamblaje del modelo 2 en la primera estación es de 5 minutos.
El siguiente operador (estación 2) comienza a trabajar con la cuarta pieza en la unidad 7, termina su trabajo en la unidad 15 (que es su frontera a la derecha) para volver a trabajar con la pieza 5. El tiempo de ensamblaje del modelo 2 en la estación 2 es de 10 minutos, por lo tanto faltan dos minutos para terminar el trabajo. Este tiempo es el utility-time. El operador de la tercera estación no puede empezar antes de la unidad 17 y tiene que esperar 1 minuto. La cuarta pieza se termina justo en 24 metros y el operador de la tercera estación tiene que recorrer 8 unidades hacia arriba para comenzar en la unidad 15 con la pieza 5 (modelo 3).

7. DESAFIOS Y PROBLEMAS

El problema general de este algoritmo de enumeración es que si no hay grandes diferencias entre los caminos, el programa calcula casi todo el árbol. Para poder calcular las longitudes de las estaciones el programa de Saber y nuestro trabajo simplifican el problema de ensamblaje (con números decimales) y consideran solo números enteros.

Las fórmulas basadas en las publicaciones de Sarker and Pan, 1997, y Sarker and Pan, 2001, tenían pequeños errores y primero hubieron de ser corregidos. Una vez faltaban unos corchetes, otra vez un índice era erróneo. Con la implementación de las formulas se tiene que tener mucho cuidado, ya que las formulas de la primera estación hacen cambiar las fórmulas de las siguientes estaciones.

Errores en Sarker and Pan, 1997:

- En pagina-7 en ecuación-6 faltan los corchetes en el cálculo del Idle-time.

$$I_{i,j+1} = MAX\left\{0, \lambda - \left[\frac{Z_{i,j} - Z_1^0}{v_c} + \sum_{m=1}^{M}(X_{m,j} \cdot t_{m,i}) - u_{i,j}\right]\right\}$$

- En pagina-8 en ecuación-11 falta el término $(\lambda \cdot v_c)$ para poder calcular correctamente el "Upstream-Distance".

Errores en Sarker and Pan, 2001:

- En pagina-7 en ecuación (12) los índices son mal. En vez de ($Z_{i+1,j}$) se encuentra ($Z_{i,j+1}$).

8. CONCLUSIÓN

El Algoritmo de enumeración es muy útil si por ejemplo el tiempo de ensamblaje de los distintos modelos varía bastante. Así se puede eliminar muchas ramas. Si el problema tiene pocos niveles pero un ancho bastante grande, se tiene que calcular muchas soluciones, a veces más de lo necesario. Aumentando el número de estaciones de la línea hasta la dimensión que se puede encontrar en la mayoría de industrias, el tiempo de cálculo crece enormemente, no siendo aconsejable utilizar este método sin adaptación.

El algoritmo llega a la solución óptima, pero en problemas de gran tamaño necesita valores acotados de entrada y una buena heurística para encontrar una buena solución inicial. También se puede asignar un factor (por ejemplo 0.9) al coste cuando se compara el coste de un nodo con el coste inicial para poder eliminar más ramas.

De todos modos en la realidad una solución buena muchas veces es suficiente. No tiene que ser la óptima que implica cálculos extensos. Una heurística puede llegar a una solución aceptable con menos tiempo de cálculo.

En este trabajo además se presentó un método rápido para conseguir todas las secuencias posibles, utilizando un sistema numérico con la base igual al número de piezas a producir.

9. BIBLIOGRAFIA:

S.Kim and B.Jeong, 2000, "Product sequencing problem in Mixed-Model Assembly line to minimize unfinished work", Department of Industrial Systems Engineering, Yonsei University, Seoul, 120-749, South Korea, The 27th Conference on Computer & Industrial Engineering, Beijing

B.R.Sarker and H.Pan, 1997, "Designing a mixed-Model assembly line to minimize the costs of idle and utility times", Department of Industrial & Manufacturing Systems Engineering, Louisiana State University, Baton Rouge, LA, USA

B.R.Sarker and H.Pan, 2001, "Designing a mixed-model, open-station assembly line using mixed-integer programming", Louisiana State University, USA and A-Plus Manufacturing Corporation, San Jose, CA, USA

T.Korkmazel and S.Meral, 2001, "Bicriteria sequencing methods for the mixed-model assembly line in just-in-time production systems", European Journal of Operational Research 131, 188-207